Модульное тестирование на базе JUnit5. Основы тестирования

Рассматриваемые вопросы

1. Основы тестирования
2. Обзор JUnit5 и Mockito

Задачи и цели модульного тестирования

- Поиск и документирование несоответствий требованиям
- Поддержка разработки и рефакторинга низкоуровневой архитектуры системы и межмодульного взаимодействия
- Поддержка рефакторинга модулей
- Поддержка устранения дефектов и отладки

Понятие модуля и его границ

- модуль - это часть программного кода, выполняющая одну функцию с точки зрения функциональных требований;
- модуль - это программный модуль, т.е. минимальный компилируемый элемент программной системы;
- модуль - это задача в списке задач проекта (с точки зрения его менеджера);
- модуль - это участок кода, который может уместиться на одном экране или одном листе бумаги;

Тестирование классов

Процесс тестирования классов как модулей иногда называют компонентным тестированием

Определение ошибок при тестировании

- дефекты инкапсуляции
- дефекты наследования
- дефекты полиморфизма
- дефекты инстанцирования

Организация модульного тестирования

1. Фаза планирования тестирования
- Этап планирования основных подходов к тестированию, ресурсное планирование и календарное планирование
- Этап определения свойств, подлежащих тестированию
- Этап уточнения основного плана, сформированного на этапе

2. Фаза получения набора тестов
- Этап разработки набора тестов
- Этап реализации уточненного плана

3. Фаза измерений тестируемого модуля
- Этап выполнения тестовых процедур
- Этап определения достаточности тестирования
- Этап оценки результатов тестирования и тестируемого модуля.

JUnit5

JUnit — библиотека для модульного тестирования программ Java.

JUnit – это Java фреймворк для тестирования, т. е. тестирования отдельных участков кода, например, методов или классов.

JUnit5

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-engine</artifactId>
    <version>5.1.0</version>
    <scope>test</scope>
</dependency>
```

JUnit5

```
import static org.junit.jupiter.api.Assertions.assertEquals;
import example.util.Calculator;
import org.junit.jupiter.api.Test;

class MyFirstJUnitJupiterTests {

    private final Calculator calculator = new Calculator();

    @Test
    void addition() {
        assertEquals(2, calculator.add(1, 1));
    }

}
```

JUnit5

```
@BeforeAll
static void initAll() {}

@BeforeEach
void init() {}

@Test
void succeedingTest() {}

@Test
void failingTest() {
    fail("a failing test");
}

@Test
@Disabled("for demonstration purposes")
void skippedTest() {        // not executed    }

@Test
void abortedTest() {
    assumeTrue("abc".contains("Z"));
    fail("test should have been aborted");
}

@AfterEach
void tearDown() { }

@AfterAll
static void tearDownAll() { }
```

Утверждения в тестах (Assertions)

- org.junit.jupiter.api.Assertions.assertEquals;
- org.junit.jupiter.api.Assertions.assertNotNull;
- org.junit.jupiter.api.Assertions.assertNull;
- org.junit.jupiter.api.Assertions.assertThrows;
- org.junit.jupiter.api.Assertions.assertTrue;
- org.junit.jupiter.api.Assertions.assertFalse;